Anonymous class :

```
Type variableName = new Type() {
    // Method overrides
};
```

Lambda Expressions Syntax

(☆ Note that Lamba Expression can only be used when the interface has one abstract method.
Also, the lamba expression cannot access the variables outside its scope! So if you need to keep track of them, anonymous classes are better in these cases)

Void in, **Void** out

```
public interface Runnable {
    public void run();
}
```

```
Runnable r1 = () -> System.out.println("line");

Runnable r2 = () -> { System.out.println("line1");
                     System.out.println("line2"); };
```

1 **Value** in, **Void** out

```
public interface Consumer {
    public void consume(double val1);
}
```

```
Consumer c1 = (x) -> System.out.println(x);
```

```
Consumer c2 = (x) -> { double z = x*x;
                       System.out.println(z); };
```

1 Value in, Value out

```
public interface Function {
    public double apply(double val1);
}
```

```
Function f1 = (y) -> y*y;
```

```
Function f2 = (y) -> { double z = y*y;
                       System.out.println(z);
                       return z; };
```

2 Values in, Value out

```
public interface BiFunction {
    public double apply(double val1, double val2);
}
```

```
BiFunction b1 = (x,y) -> x*y;

BiFunction b2 = (x,y) -> { double z = x*y;
                           System.out.println(z);
                           return z; };
```